Open access • Proceedings Article • DOI:10.1109/WACV.2015.151

Sparse Flow: Sparse Matching for Small to Large Displacement Optical Flow

Radu Timofte, Luc Van Gool

Institutions: Katholieke Universiteit Leuven

Published on: 05 Jan 2015 - Workshop on Applications of Computer Vision

Topics: Sparse approximation, Optical flow, Pixel and Blossom algorithm

SparseFlow: Sparse Matching for Small to Large Displacement Optical Flow

Radu Timofte
CVL, D-ITET, ETH Zürich, Switzerland
radu.timofte@vision.ee.ethz.ch

Luc Van Gool
VISICS, ESAT, KU Leuven, Belgium
luc.vangool@vision.ee.ethz.ch

Abstract

Despite recent advances, the extraction of optical flow with large displacements is still challenging for state-of-the-art methods. The approaches that are the most successful at handling large displacements blend sparse correspondences from a matching algorithm with an optimization that refines the optical flow. We follow the scheme of DeepFlow [33]. We first extract sparse pixel correspondences by means of a matching procedure and then apply a variational approach to obtain a refined optical flow. In our approach, coined 'SparseFlow', the novelty lies in the matching. This uses an efficient sparse decomposition of a pixel's surrounding patch as a linear sum of those found around candidate corresponding pixels. As matching pixel the one dominating the decomposition is chosen. The pixel pairs matching in both directions, i.e. *in a forward-backward fashion, are used as guiding points in the variational approach. SparseFlow is competitive on standard optical flow benchmarks with large displacements, while showing excellent performance for small and medium displacements. Moreover, it is fast in comparison to methods with a similar performance.*

1. Introduction

There is an ever-increasing amount of video content that computer vision algorithms ought to analyze. Optical flow often is an important component thereof. A robust optical flow algorithm should cope with a wide variety of conditions. These include: discontinuities (outliers, occlusions, motion discontinuities), appearance changes (illumination, chromacity, deformations), and large displacements. While we have efficient approaches for the first two issues [6, 24], how to handle large displacements to a large extent still is an open problem, despite the recent endeavors [35, 27, 9, 37, 33, 7, 18].

The seminal work of Brox and Malik [9] shows that a variational approach can better handle large displacements when a descriptor matching term is added. The idea is to guide the variational optical flow estimation by providing (sparse) correspondences from the descriptor matcher. The advantage of descriptor matching is that it can overcome arbitrarily large displacements, a strength thus incorporated into the variational optical flow methods.

Most current matching approaches are based on descriptors with a square support (*e.g.* HOGs [12]), that are invariant only to similarities. Yet, exactly under the conditions where large displacements need to be bridged, this level of invariance may be insufficient [9]. Weinzaepfel *et al.* [33] improve the descriptor matching by not only increasing the density of matched points, but also by catering for deformable matching. Their 'deep matching' solution is inspired by deep convolutional nets [17], has 6 layers, and interleaves convolutions and max-pooling.

We propose a novel matching process that is inspired by compressed sensing [13]. Thus, we work under a sparsity assumption. The pixels are described by their surrounding blocks of pixel intensities. A pixel can then be sparsely decomposed over a pool of pixels from a target image. This sparse decomposition formulation is able to cope with high image corruptions and deformations as shown by Wright *et al.* [36] for face recognition. The dominant pixel in the decomposition is likely to be the correspondence in the target image. We call this process of sparse coding and correspondence selection *sparse matching*.

We make two main contributions:

1. *robust correspondence matching*: we introduce a descriptor matching algorithm, namely **sparse matching**, able to robustly cope with image deformations and to provide highly accurate matches (precision $\sim 96\%$);

2. *small to large displacement optical flow*: our variational optical flow methods (**SparseFlow**, **SparseFlowFused**) inherit the precision and robustness to large displacements offered by sparse matching, providing top performance on MPI-Sintel dataset [11] and KITTI dataset [14].

The remainder is organized as follows. First, we review recent related work in Section 2. Then we introduce the sparse matching algorithm in Section 3. Section 4 describes

our variational optical flow approach. We present experimental results in Section 5, to conclude the paper in Section 6.

2. Related work

Large displacement in optical flow estimation. The state-of-the-art in optical flow is represented by the variational methods. The seminal work of Horn and Schunck [15] has been improved repeatedly over the years [25, 6, 10, 24, 34, 28, 4, 32]. Brox *et al.* [8] combine many of these improvements into a variational approach. The problem is formulated as an energy minimization represented by Euler-Lagrange equations, finally reduced to solving a sequence of large and structured linear systems.

Brox and Malik [9] propose to incorporate a descriptor matching component into the variational approach. Unfortunately, the local descriptors are locally rigid and reliable only at salient locations, and the matching has a pixel level precision. Adding the matching component to the variational formulation can harm the performance, especially in places with small displacements and for wrongfully proposed matches. In the context of scene correspondence, the SIFT-flow [19] and PatchMatch [5] algorithms use descriptors or small patches. Xu *et al.*[37] combines SIFT [21] and PatchMatch [5] matching for refined flow level initialization with excellent performance at the expense of computation costs. Leordeanu *et al.* [18] extend coarse matching to dense matching by enforcing affine constraints, followed by variational flow refinement. Weinzaepfel *et al.* [33] propose dense correspondences matching by means of interleaved convolutions and max-pooling layered operations, followed, again, by variational refinement. We propose 'sparse matching' for reliable and accurate pixel correspondences extraction under strong corruptions and deformations in combination with a variational flow refinement.

Descriptor matching. Extraction of local descriptors and matching are the two steps usually employed in matching images. While, initially, the descriptors of choice were extracted sparsely, invariant under scaling or affine transformations [23], the recent trend in optical flow estimation, is to densely extract rigid (square) descriptors from local frames [31, 9, 19]. The descriptor matching is usually reduced to a (reciprocal) nearest neighbor operation [21, 5, 9]. Important exceptions are the recent works of Leordeanu *et al.* [18] (enforcing affine constraints) and Weinzaepfel *et al.* [33] (non-rigid matching inspired by deep convolutional nets). We show that (i) extraction of rigid descriptors (somehow complementary to Weinzaepfel *et al.* [33]) and (ii) quasi-dense 'sparse matching' yield robust performance, with top results on MP-Sintel [11] and KITTI [14] datasets.

Sparse coding. Our proposed matching algorithm, called *sparse matching*, is based on compressed sensing theory [13]. In the context of visual vocabularies, it also shares similarities to the soft assignment procedures based on sparse coding [20]. While in the soft assignment one sample is matched to multiple 'correspondences' with weights obtained through sparse decomposition, in our case one pixel is a assigned to a single correspondence as the dominant pixel in the sparse decomposition over the target image pixels. Furthermore, we are the first to blend sparse coding into the optical flow variational estimation framework.

3. Sparse Matching

In this section we introduce the sparse matching approach to correspondence search and discuss its main features.

3.1. Insights into the approach

In compressed sensing [13] one key idea is that most of the signals admit a sparse decomposition over a mix of signals from some pool. The sparsity principle reached popularity in the vision community as sparse coding with visual word vocabularies [20]. Also, in face recognition, sparse representations had quite some impact [36]. In this case the class label is transferred from the decomposing signals to the decomposed one.

In the same vein, we see the 2D image as a collection of local patches. In particular, the image is considered a collection of textural segments, each represented by local image patches. The textural segments define subspaces that can be spanned by just a small group of their local image patches. In order to find a match for a patch, the standard is to take the closest patch disregarding the relation with the other patches. We also use the other patches, as they help at selecting the appropriate textural subspaces. The power of each local patch is augmented by the other local patches to better generalize, i.e. to predict new patches from the same textural category. A new image patch is linearly decomposed over the image pool of local patches. The sparsity guides the decomposition towards the relevant textural subspaces. Having this decomposition, we can identify the textural subspace it probably belongs to, and which patch in the pool contributes most. It is the latter that we choose as possible correspondence for the initial image patch. It is identified as the patch in the decomposition with the highest coefficient magnitude. We refer to the above process of correspondence selection as *sparse matching*.

Final *correspondences* are selected as patch pairs for which the sparse matching works both ways, that is, each patch is the sparse match of the other in a bijection.

The next section fixes the remaining open issues, like the patch descriptors, the solvers to be used, etc.

3.2. Local image patches

We need pixel-to-pixel correspondences between two subsequent images for the optical flow task at hand. To that

end, pixels are to be described by features that allow for a linear decomposition as proposed earlier. In accordance with the patch idea, we extract the features from a square image neighbourhood centered around the pixel. This patch of pixels is of fixed size. We decouple the raw RGB data of the pixels into chromacity and luminance. The block of pixel luminance values is vectorized and l_2-normalized. Such scaling adds robustness against noise but preserves the linearity underlying the decomposition. As colour helps solving ambiguities, its information is added as the mean over the patch of the R, G, and B color channels. As feature entries we use those 3 entries, multiplied by the number of patch pixels N and by a factor β which sets the importance of the average chromacity with respect to the luminance. Especially in image regions with repetitive textures it is useful to bias the patch matching towards patches with similar positions. To that end, we include in the patch descriptor vector also the 2D coordinates of the patch center. Those coordinates are first mapped to [0,1], and again multiply by N and by a factor γ allowing to set their importance in the descriptor. Thus, the feature vector for a local image patch of N pixels around the pixel $\mathbf{p}$ is as follows:

$$\mathbf{f_p} = [i_1, i_2, \cdots, i_N, \beta Nr, \beta Ng, \beta Nb, \gamma Nx, \gamma Ny]; \quad (1)$$

where i_j are the N pixel intensities [0,255], (r, g, b) are the mean pixel RGB color channels [0,255], (x, y) are the normalized [0,1] image coordinates of $\mathbf{p}$ and β and γ are scalar parameters.

Even the best of descriptors would find it difficult to steer towards the right correspondence. Some patches, *e.g.* those with homogeneous RGB-values withstand such efforts. Therefore our approach only considers patches around corners, as proposed many times before. We use Harris corners. A side effect is an acceleration of the patch matching part, which is now confined to a strongly limited number of patches.

3.3. Formulations and solvers

In the next we review a number of known robust techniques aiming at reducing the residue between an input sample and a sparse linear decomposition over a pool of samples. Based on the obtained sparse linear decomposition we decide the 'sparse match' as the sample in the decomposition with the highest importance (coefficient magnitude) for the input. Therefore, our sparse matching procedure is defined by the sparse decomposition technique employed.

Nearest Neighbor (NN) is the standard approach for determining the match, $\mathbf{x}_a$, for a query $\mathbf{y}$ from a pool of M samples $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_M]$, where:

$$a = \arg\min_i \|\mathbf{y} - \mathbf{x}_i\|_2. \quad (2)$$

Besides NN, for our sparse matching procedure we consider three other sparse linear decomposition methods.

Sparse Representation (SR) [36] enforces the sparsity in the decomposition by l_1-regularizing the least squares formulation:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_1, \quad (3)$$

where $\mathbf{y}$ is the query (pixel feature in our case), $\mathbf{X}$ the pool, $\mathbf{w}$ are the coefficients, λ is the regulatory parameter.

The Iterative Nearest Neighbors (INN) method [29, 30] combines the power of SR with the computational simplicity of NN by means of a constrained decomposition:

$$\{\hat{\mathbf{s}}\}_{i=1}^K = \arg\min_{\{\mathbf{s}\}_{i=1}^K} \|\mathbf{y} - \sum_{i=1}^K \frac{\lambda}{(1+\lambda)^i}\mathbf{s}_i\|_2 \quad (4)$$

where λ is the regulatory parameter, $\hat{\mathbf{s}}_i$ are samples selected from $\mathbf{X}$, and the imposed weights $\frac{\lambda}{(1+\lambda)^i}$ sum up to 1 for $K \to \infty$. $\mathbf{x}_j \in \mathbf{X}$ may be selected multiple times in (4), therefore its coefficient $\hat{w}_j$ in the INN representation $\hat{\mathbf{w}}$ is

$$\hat{w}_j = \sum_{i=1}^K \frac{\lambda}{(1+\lambda)^i}[\mathbf{x}_j = \hat{\mathbf{s}}_i] \quad (5)$$

where $[\cdot = \cdot]$ is 1 for equality and 0 otherwise. As shown in [30], with a tolerance $\theta = 0.05$ we recover the coefficients up to 0.95 and need K NN iterations of the INN algorithm, where

$$K = \lceil -\frac{\log(1-\theta)}{\log(1+\lambda)} \rceil. \quad (6)$$

Locally Linear Embedding (LLE) [26] encodes a sample $\mathbf{y}$ over its neighborhood $\mathcal{N}_{\mathbf{X}}(\mathbf{y})$ of size K in $\mathbf{X}$, $\mathbf{z}_i \in \mathcal{N}_{\mathbf{X}}(\mathbf{y})$. The sparsity is directly imposed by restricting the decomposition to the local neighborhood of samples.

$$\hat{\mathbf{c}} = \arg\min_{\mathbf{c}} \|\mathbf{y} - \sum_{i=1}^K c_i\mathbf{z}_i\|_2 \quad \text{subject to:} \sum_{j=1}^K c_j = 1 \quad (7)$$

In this case, $\hat{\mathbf{w}}$ represents the coefficients in the LLE representation over $\mathbf{X}$, where the nonzero coefficients are those given by $\hat{\mathbf{c}}$ corresponding to the K nearest neighbors.

For all the decomposition methods above we normalize $\tilde{\mathbf{w}} = |\hat{\mathbf{w}}|/\|\hat{\mathbf{w}}\|_1$. The most important sample (the 'match'), $\mathbf{x}_a$, in the representation of $\mathbf{y}$ over $\mathbf{X}$ is the one with the largest coefficient magnitude:

$$a = \arg\max_i \tilde{w}_i. \quad (8)$$

When applied to two images, the pixels (features) of the first image are decomposed over the pixels (features) from the second image. The matches are selected in the second image. Then, the process is repeated and matches are found for the pixels from the second image in the first. The reciprocal matched pixels are correspondences we trust and the result of our sparse matching process. The score of a correspondence is taken as the average of the corresponding normalized coefficients of the two pixel decompositions.

4. SparseFlow

Our optical flow method, called 'SparseFlow', uses our proposed sparse matching. The best solver to use will follow from the experiments in the next section. It also builds on the variational optical flow strategy as expounded in the DeepFlow paper [33]. This variational approach differs from that of Brox and Malik [9] by the incorporation of the external matching component, the addition of a normalization in the data term (to reduce the impact of areas with high image derivatives), and the use of a different weight at each level. For details on those aspects, we refer the reader to the original work [33].

5. Experiments

In this section we evaluate the SparseFlow approach. After introducing the benchmarks, we show how sparse matching compares with other matching algorithms, how the matching impacts the flow estimation of the variational optimization, and finally we report on the SparseFlow results. Our codes are publicly available at:
`http://www.vision.ee.ethz.ch/~timofter/`

5.1. Datasets

The MPI-Sintel dataset [11, 2] is a benchmark with long video game sequences, large motions ($\sim 10\%$ of displacements are larger than 40 pixels in the training data), and many (rendered) image degradations such as blur or reflections. We focus on the 'Final' version of the dataset. As in [33] we randomly split the original training set into a training set (20%) and validation set (80%). The flow performance is quantified using the endpoint errors (EPE). 'EPE all' stands for average EPE over all pixels, while s10-40 only for those with motions ranging from 10 to 40 pixels, and similarly for s0-10 or s40+.

The KITTI dataset [14, 1] contains real-world sequences captured from a driving platform. This dataset exhibits a large number of real challenging conditions. About 16% of the pixel motions are over 20 pixels.

5.2. Parameters and Sparse Matching

The performance of the sparse matching strategy is influenced by the choice of parameters. Hence, first we discuss the impact of the features, regulatory parameters and matching decisions on its overall performance. For setting the parameters we use our small training set from MPI-Sintel. We found it useful to cope with the scale changes explicitly, by considering both the pixels from the original image and its half resolution version. After trying different combinations of parameters, we finally fixed their values to the following settings for all our experiments: patch sizes of 13×13 (thus, $N = 169$), the color parameter $\beta = 0.33$ and the coordinates parameter $\gamma = 0.01$. In order to reduce the number of patches to corners, we use Kovesi's Harris corner detector function [16] with the following parameters: sigma set to 1, radius to 2, and the minimum corner score to 1. In this way, we extract only a few thousands pixel descriptors per image.

For obtaining the linear decomposition we considered SR (with the lasso solver from SPAMS library [22]), INN (Matlab solver provided by the authors [30]), and LLE (with Matlab codes based on [26]). Our choice of parameters is $\lambda = 0.1$ for SR, $\lambda = 0.25$ for INN, and 7 the number of nearest neighbors for LLE. The performance and the running times of our matching solvers where comparable even if SR uses C++ code whereas INN and LLE use Matlab scripts. The LLE matching is faster than INN with our settings but the final flow performance was slightly below that with INN. Therefore, we use INN in all our further experiments.

We drop correspondences whenever the average of their normalised coefficients for the two-ways decompositions is below 0.5. For the MPI Sintel validation set the average number of correspondences found by the sparse matching approach is 2330. This is larger than the average number 1797 as obtained by the deep matching approach (correspondences provided by the authors).

We decided to keep the parameters for the variational approach that were already used in the DeepFlow paper [33]. Indeed, for SparseFlow we use the settings of the variational component tuned for DeepFlow best performance, and therefore are potentially suboptimal for our SparseFlow approach. Nevertheless, this allows for direct comparison in flow performance between our sparse matching and flow (SparseFlow) approach and the deep matching and flow (DeepFlow) approach. Moreover, a fused approach (SparseFlowFused) for matching and flow is easily derived since the approaches share the variational component with the same parameters.

We brought the strengths of our sparse matches to the range of values of the deep matching. To that end, we rescore our sparse matches using the DeepFlow rescoring script. The 'SparseFlowFused' approach that is referred to in the further discussion results from using both the rescored sparse and deep matches in the same variational optimization.

5.3. Comparison of matching algorithms

We compare our sparse matching directly with the deep matching code provided by its authors [33], and indirectly with diverse state-of-the-art algorithms: KLT tracks [3], sparse SIFT matching [21] (here SIFT-NN), dense HOG matching with uniqueness as in LDOF [9] (here HOG-NN). For the quantitative results as we report them, we adhere to the setup proposed by Weinzaepfel *et al.* [33]. We impose a fixed grid with a spacing of 15 pixels. The percentage of

Table 1. Evaluation of the matching methods on the 'Final' MPI-Sintel validation set. We report our results and the results from [33] for a *different* validation set split.

Matching input	Precision	Density	EPE all	s0-10	s10-40	s40+
Sparse+Deep (SparseFlowFused)	**94.04**%	**84.69**%	**4.317**	**0.726**	**4.968**	**28.514**
Sparse matching (SparseFlow)	**95.65**%	43.91%	4.872	**0.732**	**4.756**	34.665
Deep matching (DeepFlow)	91.95%	**80.57**%	**4.592**	0.922	5.407	**28.991**
Deep matching (DeepFlow)[33]	92.07%	**80.35**%	**4.422**	**0.712**	5.092	**29.229**
HOG-NN [33]	92.49%	40.06%	5.273	0.764	**4.972**	37.858
SIFT-NN [33]	**93.89**%	16.35%	5.444	0.846	5.313	38.283
KLT [33]	91.25%	35.61%	5.513	0.820	5.304	39.197
No match [33]	–	–	5.538	0.786	5.229	39.862

grid points with at least one match within its surrounding 15×15 cell yields a density measure. The percentage of matches with an error smaller than 10 pixels gives a precision measure.

Table 1 summarizes our sparse matching results (with our validation set split) and the results reported by [33] (with their validation set). Our sparse matching method improves precision over that of DeepFlow (95.65% vs. 91.95%), albeit at a substantially lower density. A side note is due at this point, as deep matching extracts points over a dense grid and is designed for providing grid-dense matches, whereas the other methods are saliency and/or texture driven. See Fig. 1 for some visual results.

Combining the sparse and deep matching correspondences leads to the best density (84.69%), while the precision (94.04%) is still better than that of the deep matching alone. The sparse matching is significantly faster than the deep matching, therefore the combination implies improvement in both precision and density over deep matches alone at the price of a small increase in computation time ($\sim$ 1s in Table 2).

5.4. Impact of the matches on the flow

In order to assess the impact of the matches on the flow estimation, we compare all the matching methods from the previous section, this time as matching term in the variational optimization. Note that our SparseFlow and SparseFlowFused methods use the same variational optimization codes as [33]. Sparse matching corresponds to the SparseFlow, deep matching to the DeepFlow, sparse+deep matching to SparseFlowFused method.

Table 1 summarizes the flow performance of the variational optimization in terms of average endpoint error (EPE) on our MPI-Sintel validation set and on Weinzaepfel *et al.*'s validation set split, as reported in [33]. We note here that adding matches to the variational optimization is beneficial. Sparse matching (SparseFlow) outperforms the other methods especially for small (s0-s10) and medium (s10-s40) displacements. At s40+ the error with deep matching (DeepFlow) is more than 5 pixels smaller than the one with sparse matching, while this one is 3 up to 5 pixels smaller than those obtained with the other reported methods. This suggests the importance of a good density of the matches for good optical flow performance. Sparse matching based flow performance is in between deep matching and the other matching methods. The combination of sparse and deep matching (SparseFlowFused) leads to the best flow performance, an ~ 0.2 average EPE improvement over any individual result. Also, there is a gap of ~ 0.4 average EPE between the flow results with sparse matches and the other results using rigid descriptor formulations. Fig. 1 compares results obtained with the sparse and the dense matching approaches. Deep matching covers the image space clearly better, its average density being twice higher than that of sparse matching, as shown in Table 1. Deep matching provides matches also in flat areas and textureless areas, it is guided by an uniformely spaced grid. Yet sparse matching is not contraint to a grid and often provides denser matches in critical areas, such as textured areas and/or with pronounced edges. Sparse matching provides overall higher precision and density in edgy areas, this leading to the best performance of SoftFlow for small to medium (s0-s40) displacements, and reasonably good for large displacements. On the other hand, deep matching while less precised, has an overall high density also in textureless areas, and DeepFlow benefits from this especially in the estimation of large (s40+) displacements. By combining soft and deep matches we achieve strong overall performance from small to large displacements, at the price of computing the extra matches ($\sim$ 1s for the sparse matches).

Table 2. Results on the 'Final' MPI-Sintel test set. For more results, see the MPI-Sintel website [2].

Method	EPE all	s0-10	s10-40	s40+	time[s]
SparseFlowFused	**7.189**	1.275	3.963	44.319	20
SparseFlow	7.851	**1.071**	**3.771**	51.353	**10**
DeepFlow [33]	7.212	1.284	4.107	**44.118**	19
S2D-Matching[18]	7.872	1.172	4.695	48.782	$\sim$2000
MDP-Flow2 [37]	8.445	1.420	5.449	50.507	709
Data-Flow [32]	8.868	1.794	5.294	52.635	180
LDOF [9]	9.116	1.485	4.839	57.296	30
Classic+NL [28]	9.153	1.113	4.496	60.291	301

5.5. Results on MPI-Sintel

Table 2 compares our results with state-of-the-art results for the MPI-Sintel test set. Some visual results for our

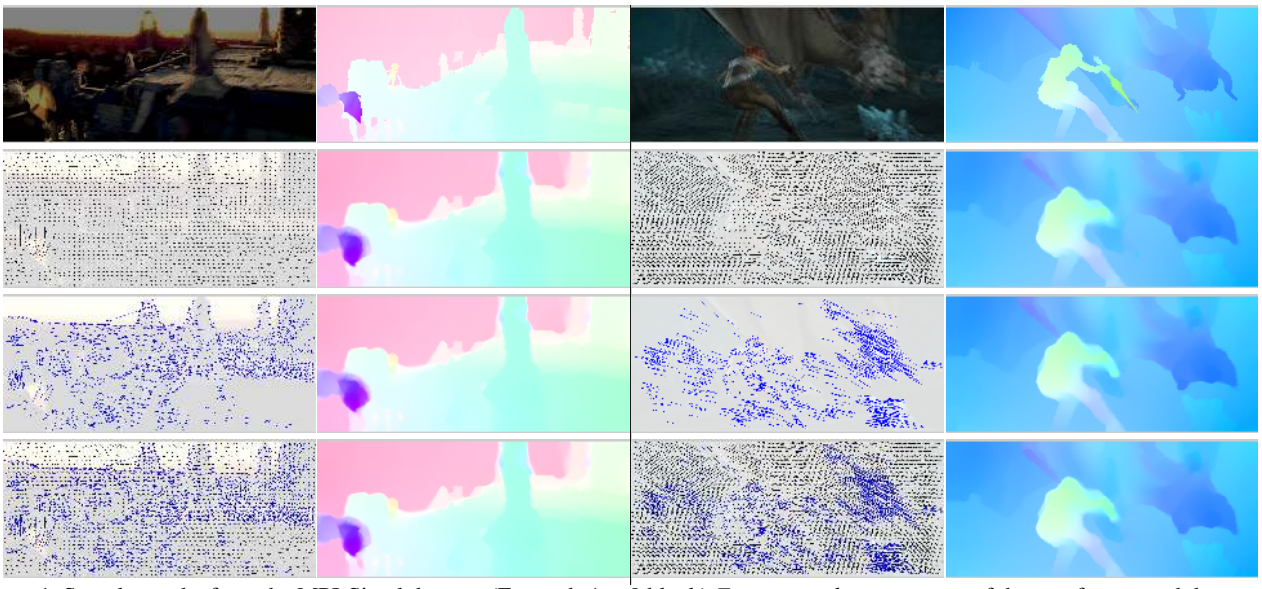

Figure 1. Sample results from the MPI-Sintel dataset. (For each 4×2 block) *From top to bottom:* mean of the two frames and the ground truth flow; deep matching and flow computed with DeepFlow [33]; our sparse matching and flow (SparseFlow); the combined sparse and deep matches and the flow (SparseFlowFused). More flow results are available on MPI-Sintel website [2].

Table 3. KITTI results. More results are available on the KITTI website [1].

Method	Out-Noc3	Out3	AEE-Noc	AEE	Time	Environment
SparseFlow	9.09%	19.32%	2.6	7.6	10 s	1 core @3.5GHz(Matlab+C/C++)
DeepFlow [33]	7.22%	17.79%	1.5	5.8	17 s	1 core @3.6GHz(Python+C/C++)
Data-Flow [32]	7.11%	14.57%	1.9	5.5	180 s	2 cores@2.5GHz(Matlab+C/C++)
Classic+NL[28]	10.49%	20.64%	2.8	7.2	888 s	1 core @2.5GHz (C/C++)
LDOF [9]	21.93%	31.39%	5.6	12.4	60 s	1 core @2.5GHz (C/C++)

methods (SparseFlow and SparseFlowFused) compared to DeepFlow on this dataset are shown in Fig. 1. For more results we refer the reader to the MPI-Sintel dataset website [2]. The parameters of our method were optimized on the training set. Our SparseFlow method comes second to the DeepFlow method in terms of EPE performance, but it is almost double as fast. The fused method, SparseFlowFused, achieves the best performance from the compared methods when considering its overall EPE value, while still being practically as fast as DeepFlow. We refer the reader to the official MPI-Sintel webpage for complete results [2].

SparseFlow is the best for small and medium displacements (s0-10 & s10-40), while being competitive for large displacements (s40+). Overall SparseFlowFused benefits from the superior density of the combined sparse and deep matches and their complementarity. Also, SparseFlow is the fastest method – the sparse matching is faster than the deep matching part of DeepFlow[1]. Sparse matching takes on average 1 second per image pair while the variational part takes 9 seconds. Our running time was computed on an Intel i7 4770 CPU, while the other running times are as reported in [33] and the other original works.

[1]Deep matching (v1.0) [33] takes 31s on avg on our machine (1 core) for the suggested '1024x512 iccv_settings'.

5.6. Results on KITTI

Table 3 and Fig. 2 show KITTI results. For more quantitative and qualitative results we refer to the dataset website [1, 14]. Out-Noc3 and Out3 are the percentage of pixels with an EPE over 3 pixels for non-occluded areas and for all pixels, respectively. AEE is the average endpoint error over all pixels, AEE-Noc excludes the occluded areas.

We use the same sparse matching parameters that were learned from the MPI-Sintel training set. SparseFlow provides robust performance and is computationally more efficient than the other top methods with a comparable performance. We found on the training data that there is only a small difference in performance for SparseFlow when using the variational parameters learned from MPI-Sintel vs. those learned from KITTI. Thus, for realistic scenarios, the parameters of our method seem to generalize well.

6. Conclusions

We proposed the SparseFlow algorithm to extract optical flow, incl. large displacements. It is based on the introduction of a novel sparse matching term into a variational optimization framework. The sparse matching procedure is based on the sparsity idea. It provides pixel correspondences under difficult conditions and at a low

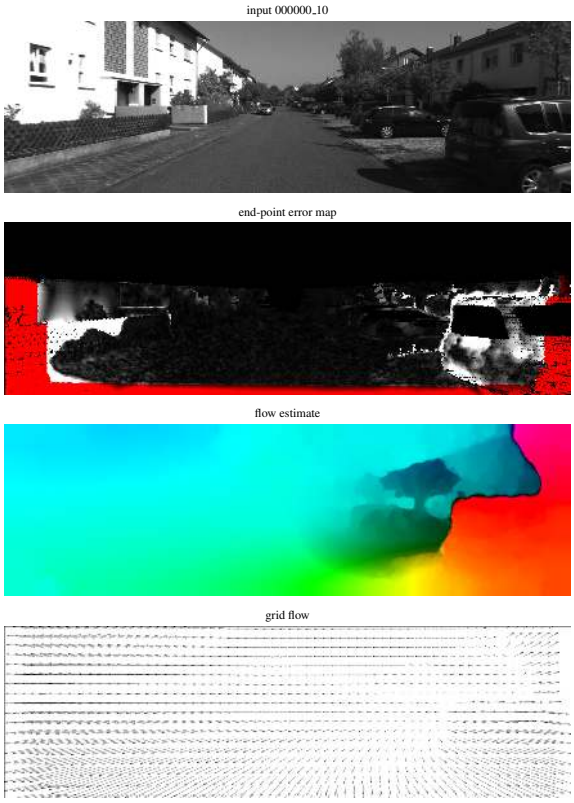

Figure 2. SparseFlow result on KITTI. More results are available on the KITTI website [1].

computational cost. As shown experimentally, SparseFlow performs excellently for small to medium motions, while being competitive for large motions. It is one of the fastest CPU methods.

Acknowledgments This was partly supported by the ETH General Founding (OK) and the European Research Council (ERC) under the project VarCity (#273940).

References

[1] KITTI website. http://www.cvlibs.net/datasets/kitti, 2014.

[2] MPI Sintel website. http://sintel.is.tue.mpg.de/results, 2014.

[3] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework. *IJCV*, 2004.

[4] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *IJCV*, 2011.

[5] C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein. The generalized PatchMatch correspondence algorithm. In *ECCV*, 2010.

[6] M. J. Black and P. Anandan. The robust estimation of multiple motions: parametric and piecewise-smooth flow fields. *CVIU*, 1996.

[7] J. Braux-Zin, R. Dupont, and A. Bartoli. A general dense image matching framework combining direct and feature-based costs. In *ICCV*, 2013.

[8] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*, 2004.

[9] T. Brox and J. Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE Trans. PAMI*, 2011.

[10] A. Bruhn and J. Weickert. Towards ultimate motion estimation: Combining highest accuracy with real-time performance. In *ICCV*, 2005.

[11] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, 2012.

[12] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[13] D. L. Donoho. Compressed sensing. *IEEE Trans. Information Theory*, 52(4):1289–1306, April 2006.

[14] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The KITTI dataset. *IJRR*, 2013.

[15] B. K. Horn and B. G. Schunck. Determining optical flow. *Proc. SPIE 0281, Techniques and Applications of Image Understanding*, 1981.

[16] P. D. Kovesi. MATLAB and Octave functions, 2014. <http://www.csse.uwa.edu.au/∼pk/research/matlabfns/>.

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[18] M. Leordeanu, A. Zanfir, and C. Sminchisescu. Locally affine sparse-to-dense matching for motion and occlusion estimation. In *ICCV*, 2013.

[19] C. Liu, J. Yuen, and A. Torralba. SIFT flow: Dense correspondence across scenes and its applications. *TPAMI*, 2011.

[20] L. Liu, L. Wang, and X. Liu. In defense of soft-assignment coding. In *ICCV*, 2011.

[21] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[22] J. Mairal, F. Bach, J. Ponce, G. Sapiro, R. Jenatton, and G. Obozinski. SPAMS: SPArse Modeling Software, v2.4. http://spams-devel.gforge.inria.fr/downloads.html, 2014.

[23] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *IJCV*, 2005.

[24] N. Papenberg, A. Bruhn, T. Brox, S. Didas, and J. Weickert. Highly accurate optic flow computation with theoretically justified warping. *IJCV*, 2006.

[25] M. Proesmans, L. Van Gool, E. Pauwels, and A. Oosterlinck. Determination of optical flow and its discontinuities using non-linear diffusion. In *ECCV*, 1994.

[26] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

[27] F. Steinbrucker, T. Pock, and D. Cremers. Large displacement optical flow computation without warping. In *ICCV*, 2009.

[28] D. Sun, S. Roth, and M. J. Black. Secrets of optical flow estimation and their principles. In *CVPR*, 2010.

[29] R. Timofte and L. Van Gool. Iterative Nearest Neighbors for classification and dimensionality reduction. In *CVPR*, 2012.

[30] R. Timofte and L. Van Gool. Iterative nearest neighbors. *Pattern Recognition*, 48:60–72, 2015.

[31] E. Tola, V. Lepetit, and P. Fua. A fast local descriptor for dense matching. In *CVPR*, 2008.

[32] C. Vogel, S. Roth, and K. Schindler. An evaluation of data costs for optical flow. In *GCPR*, 2013.

[33] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *ICCV*, 2013.

[34] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof. Anisotropic Huber-L1 optical flow. In *BMVC*, 2009.

[35] J. Wills, S. Agarwal, and S. Belongie. A feature-based approach for dense segmentation and estimation of large disparity motion. *IJCV*, 2006.

[36] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Trans. PAMI*, 2009.

[37] L. Xu, J. Jia, and Y. Matsushita. Motion detail preserving optical flow estimation. *IEEE Trans. PAMI*, 2012.